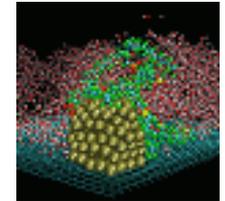
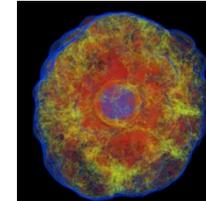
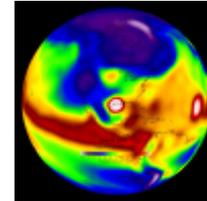
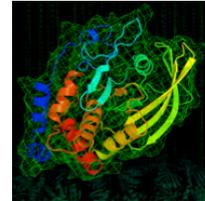
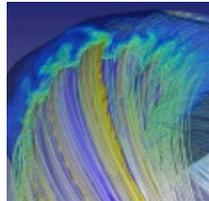
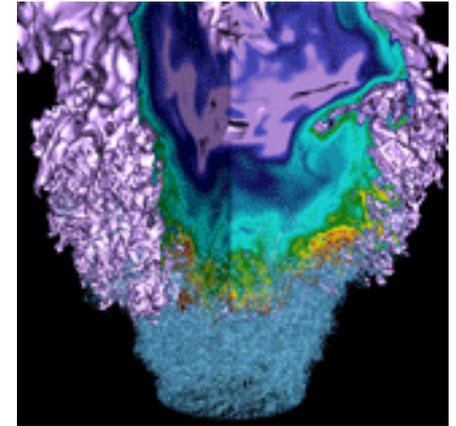


Profiling your application with Intel VTune at NERSC



- **Focus: On-node performance analysis**
 - Sampling and trace-based profiling
 - Performance counter integration
 - Memory bandwidth analysis
 - On-node parallelism: vectorization and threading
- **Pre-defined analysis experiments**
- **GUI and command-line interface (good for headless collection and later analysis)**
- **NERSC availability (as the vtune module)**
 - Edison (Dual 12-core Ivy Bridge)
 - Babbage (Dual 8-core Sandy Bridge + Dual Xeon Phi)

Running VTune on Edison I



- Use the Cray `cc` or `ftn` wrappers for the Intel compilers
- Suggested compiler flags:
 - `-g` : enable debugging symbols
 - `-O2` : use production-realistic optimization levels (not `-O0`)
- To use VTune on Edison, you have to:
 - Run within a CCM job (batch or interactive)
 - Use dynamic linking if profiling OpenMP code (`-dynamic`)
 - Use a working directory on a Lustre \$SCRATCH filesystem

```
edison09:BGW > ftn -dynamic -g -O2 -xAVX -openmp bgw.f90 -  
o bgw.x  
edison09:BGW > mkdir $SCRATCH/vtune-runs  
edison09:BGW > cp bgw.x $SCRATCH/vtune-runs/  
edison09:BGW > cd $SCRATCH/vtune-runs/  
edison09:vtune-runs > qsub -I -q ccm_int -l mppwidth=24  
wait ...
```

Running VTune on Edison II



- **Once you're in a CCM job (either interactive or batch script)**
 - cd to your submission directory
 - Launch VTune to profile your code on a compute node with aprun

```
CCM Start success, 1 of 1 responses
nid02433:~ > cd $PBS_O_WORKDIR
edison09:vtune-runs > module load vtune
nid02433:vtune-runs > aprun -n 1 ampxe-cl -collect
experiment_name -r result_dir -- ./bgw.x
```

- **ampxe-cl is the VTune CLI**
 - `-collect` : specifies the collection experiment to run
 - `-r` : specifies an output directory to save results
- **Set OMP_NUM_THREADS and associated aprun options (-d, -S, -cc depth, -cc numa_node) as needed**
- **Results can be analyzed by launching ampxe-gui and navigating to the result directory (preferably in NX)**

- **Available on Edison and Babbage (SNB + Xeon Phi)**

```
nid02433:vtune-runs > aprun -n 1 ampxe-cl -collect gener  
al-exploration -r ge_results -- ./bgw.x
```

- **Detailed characterization of relevant performance metrics throughout your application**
 - Default: low-level detail aggregated into summary metrics
 - Mouse-over for explanation of their significance
 - Can be used to characterize locality issues, poor vectorization, etc.
- **Multiple “viewpoints” available:**
 - Direct access to hardware event counters
 - Spin / sync overhead for OpenMP threaded regions

Experiments: General exploration



Elapsed Time: 60.181s

- Clockticks: 185,122,277,683
- Instructions Retired: 139,162,208,743
- CPI Rate: 1.330
The CPI may be too high. This could be caused by issues such as memory stalls, instruction starvation, branch misprediction or long latency instructions. Explore the other hardware-related metrics to identify what is causing high CPI.
- MUX Reliability: 0.989
- Paused Time: 0s

Filled Pipeline Slots:

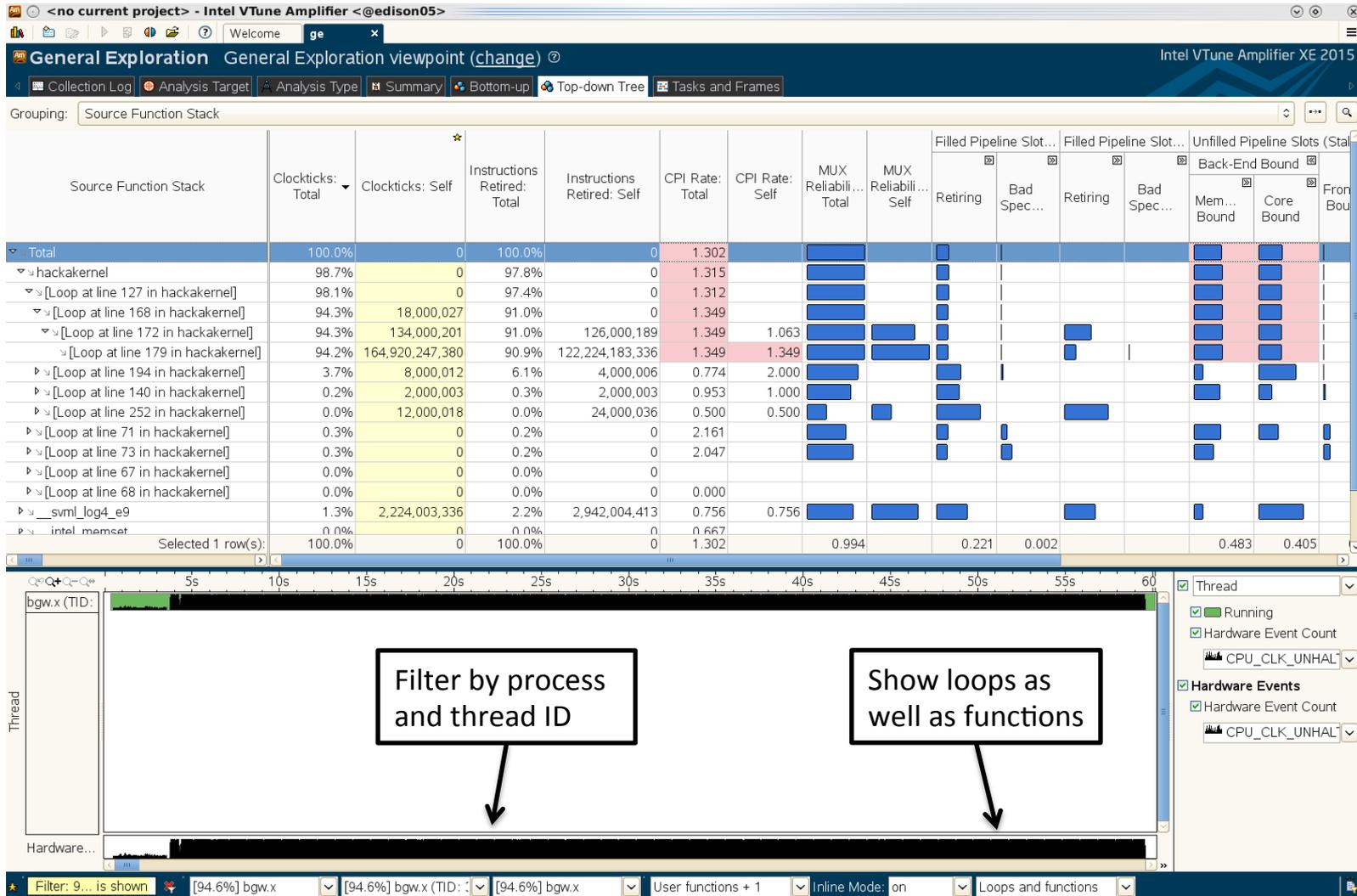
- Retiring:** 0.215
 - Microcode Sequencer: 0.004
 - General Retirement:** 0.211
 - Other: 0.663
This metric represents a non-floating-point (FP) uop fraction the CPU has executed. If your application has no FP operations, this is likely to be the biggest fraction.
 - FP Arithmetic:** 0.320
This metric represents an overall arithmetic floating-point (FP) uops fraction the CPU has executed.
 - FP x87: 0.011
 - FP Scalar: 0.000
 - FP Vector: 0.309
This metric represents an arithmetic floating-point (FP) vector uops fraction the CPU has executed. Make sure vector width is expected.
- Bad Speculation:** 0.002

Unfilled Pipeline Slots (Stalls):

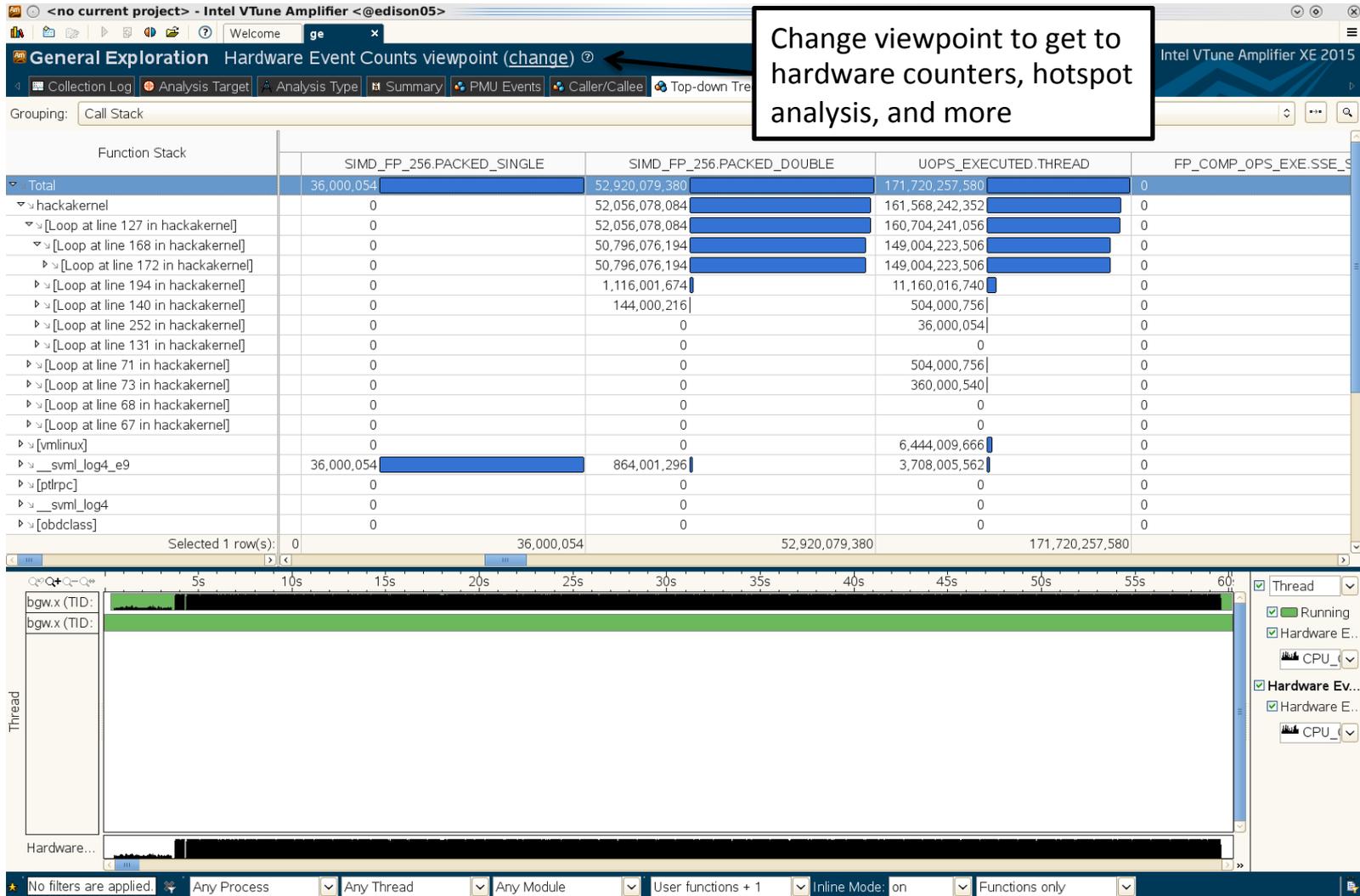
- Back-End Bound:** 0.774
Identify slots where no uOps are delivered due to a lack of required resources for accepting more uOps in the back-end of the pipeline. Back-end metrics describe a portion of the pipeline where the out-of-order scheduler dispatches ready uOps into their respective execution units, and, once completed, these uOps get retired according to program order. Stalls due to data-cache misses or stalls due to the overloaded divider unit are examples of back-end bound issues.
- Memory Bound:** 0.492
This metric shows how memory subsystem issues affect the performance. Memory Bound measures a fraction of cycles where pipeline could be stalled due to demand load or store instructions. This accounts mainly for incomplete in-flight memory demand loads that coincide with execution starvation in addition to less common cases where stores could imply back-pressure on the pipeline.
 - L1 Bound:** 0.017
 - L3 Bound:** 0.101
 - DRAM Bound:** 0.346
This metric shows how often CPU was stalled on the main memory (DRAM). Caching typically improves the latency and increases performance.
 - Memory Bandwidth: 0.262
This metric shows how often CPU could be stalled due to approaching bandwidth limits of the main memory (DRAM). Consider improving data locality in NUMA multi-socket systems.
 - Memory Latency: 0.603
This metric shows how often CPU could be stalled due to the latency of the main memory (DRAM). Consider

A whole lot of summary metrics!

Experiments: General exploration



Experiments: General exploration



Experiments: Memory bandwidth



- **Available on Edison and Babbage (Xeon Phi only)**
 - **Caveat:** avoid Babbage SNB for now (node will lock up)

```
nid02433:vtune-runs > aprun -n 1 amplxe-cl -collect bandwidth -r bw_results -- ./bgw.x
```

- **Gives DRAM read / write traffic as a function of time during program execution**
- **Useful to first calibrate with a well-understood code on the same platform (e.g. STREAM)**
- **Can help determine whether your code is at least partially (effectively) BW bound**

Experiments: Bandwidth



<no current project> - Intel VTune Amplifier <@edison05>

Welcome bw.stream x

Bandwidth

Bandwidth viewpoint (change)

Intel VTune Amplifier XE 2015

Collection Log Analysis Target Analysis Type Summary Bottom-up

Elapsed Time: 73.320s

CPU Time: 827.786s
Instructions Retired: 412,366,618,549
CPI Rate: 5.621

The CPI may be too high. This could be caused by issues such as memory stalls, instruction starvation, branch misprediction or long latency instructions. Explore the other hardware-related metrics to identify what is causing high CPI.

Paused Time: 0s

Average Bandwidth

Package	Bandwidth, GB/sec	Read Bandwidth, GB/sec	Write Bandwidth, GB/sec
package_0	49.288	29.850	19.439
package_1	0.088	0.046	0.042

Average BW listed by CPU package

OpenMP Region Duration Histogram

This histogram shows the total number of region instances in your application executed with a specific duration. High number of slow instances may signal a performance bottleneck. Explore the data provided in the Bottom-up, Top-down Tree, and Timeline panes to identify code regions with the slow duration.

OpenMP Region: main\$omp\$parallel:12@unknown:168:175

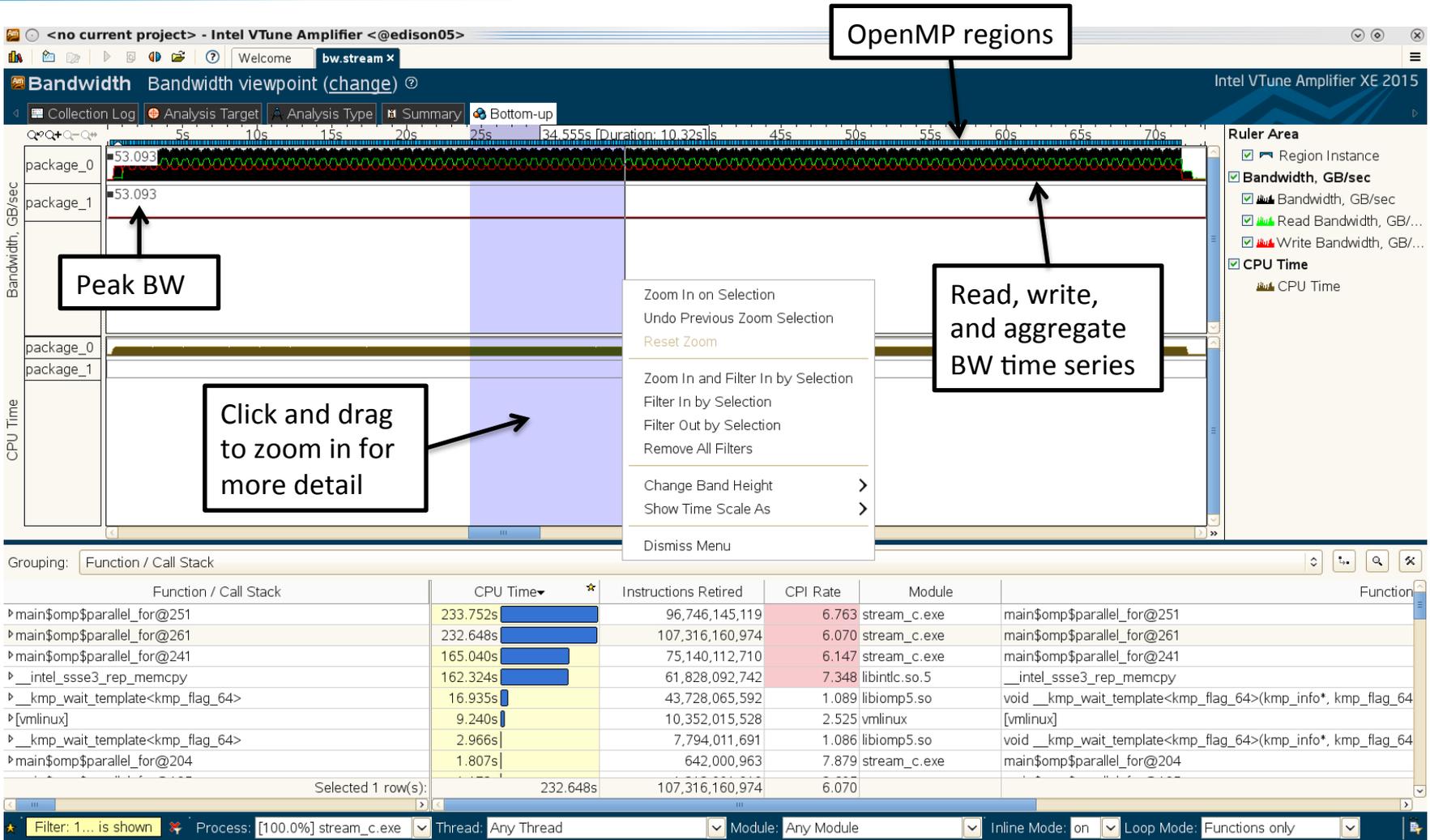
Instance Count: 1
Duration Type (sec): 0.217155
Good

Collection and Platform Info

This section provides information about this collection, including result set size and collection platform data.

Application Command Line: ./stream_c.exe
User Name: sfrench
Operating System: 3.0.101-0.31.1_1.0502.8394-cray_ari_c SUSE Linux Enterprise Server 11 (x86_64)
VERSION = 11
PATCHLEVEL = 3

Experiments: Bandwidth



- **At NERSC**

- On our debugging and profiling tools pages:
<http://www.nersc.gov/users/software/debugging-and-profiling/vtune/>
- More details on how to run your analysis on both the Edison compute nodes and the Babbage Xeon Phi
- Pointers to materials from previous NERSC trainings

- **At Intel**

- Main documentation for 2015 version:
<https://software.intel.com/en-us/node/529213>
- Detailed descriptions of the various experiment types
- Pointers to tutorials on specific topics or platforms